

# Instructions for using Microsoft Visual Studio 2015 with the CCI Configuration POC

September 13, 2016

---

## Table of Contents:

1. Introduction
  2. Prerequisites
  3. Steps
    - Step1: Build the two PoC libraries
    - Step2: Run regression tests
    - Step3: Building MSVC Projects for CCI applications
- 

## Introduction

This document describes how to use MSVS2015 with CCI. Refer to `cci/msvc10/ci/README.txt` in the github report for general Windows instructions.

Step-by-step instructions are provided for the following:

1. Building the two Proof of Concept (PoC) libraries
    - `cciapi.lib`
    - `ccibrokerimpl.lib`
  2. Running the batch mode regression test suite(`verify.pl`)
    - Consisting of both user and developer examples
  3. Building and running the examples inside Microsoft Visual Studio 2015
- 

## Prerequisites

Tested tool versions are noted below only for reference; other versions may work fine.

1. SystemCInstallation

- Download the 2.3.2 version of 'Core SystemC Language and Examples' from: <http://accelera.org/downloads/standards/systemc>.
  - Copy the msvc10 directory to msvc14 and build the SystemC library for desired configuration (e.g. x64 Debug).
2. Initialize a github workspace
    - Install github windows client <https://help.github.com/articles/set-up-git/>
    - Refer to [Getting started in the CCI WG.docx](#) for creating a fork and cloning the repository
  3. Install MSVS2015SP1
    - <https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>
  4. Install Cygwin Perl Package and Cygwin diffutils package for 64 bit Installation from <https://cygwin.com/>; the following versions were tested:
    - Cygwin 2.5.1-1
    - Perl 5.22.2-1
    - Diffutils 3.3-3
  5. Install Boost Library, version; 1.57.0 was tested
    - Follow the steps from <http://slu.livejournal.com/17395.html>
    - Update boost\config\compiler\visualc.hpp to support latest version of MSVS2015 SP1. The highlighted change below is required to avoid the 'Unknown Compiler version' warnings. The current version of MSVS2015 (SP1) we are using is Microsoft (R) C/C++ Optimizing Compiler Version 19.00.23918 for x64.

```
// last known and checked version is 19.00.23918 (VC14 CTP4):
#if (_MSC_VER > 1800 && _MSC_FULL_VER > 190023918)
#   if defined(BOOST_ASSERT_CONFIG)
#       error "Unknown compiler version - please run the configure tests and report the results"
#   else
#       pragma message("Unknown compiler version - please run the configure tests and report the results")
#   endif
#endif
```

6. Since MSVS 2015 does not support 'gets', update systemc.h

# Initial support for MSVC 2015 #120

**Merged** pah merged 3 commits into `OSCI-WG:master` from `pah:intc-fix-msvc-2015` on Nov 3 2015

Conversation 0 Commits 3 Files changed 2

Showing changes from 1 commit 1 changed file

+11 -1 Diff options

## systemc.h: don't refer to std::gets on MSVC >= 2015

< Prev Next >

Quoting "Breaking changes in Visual C++" for MSVC 2015:

The `gets` and `_getws` functions have been removed. The `gets` function was removed from the C Standard Library in C11 because it cannot be used securely.

Reported-by: Martin Freibothe <martin.freibothe@intel.com>

Signed-off-by: Philipp A Hartmann <philipp.a.hartmann@intel.com>



pah committed on Aug 5 2015

commit 92c5308896179dbc2aed3772a1e5ac08b8cc867d

12 src/systemc.h

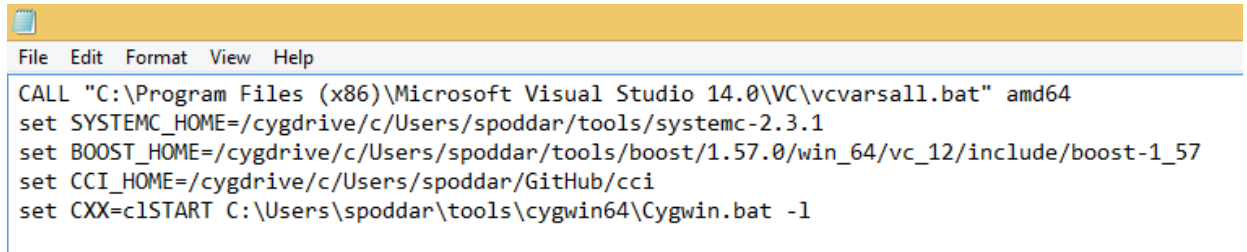
View

```
@@ -59,6 +59,12 @@
59      #include <utility>
60      #include <vector>
61
62      // C++11 deprecates std::gets, as it can't be used safely.
63      // Compilers are starting to remove it from their headers.
64      #if defined(_MSC_VER) && (_MSC_VER >= 1900)
65      #define SC_NO_STDGETS_
66      #endif
67      +
68      // USINGS FOR I/O STREAM SUPPORT:
69
70      using std::ios;
71
72      @@ -115,7 +121,9 @@
115      using std::fputs;
116      using std::getc;
117      using std::getchar;
118      #if !defined(SC_NO_STDGETS_)
119      using std::gets;
120      #endif
121      using std::putc;
122      using std::putchar;
123      using std::puts;
124
125      @@ -343,4 +351,6 @@ using namespace sc_core;
343      typedef ::std::string sc_string;
344      #endif
345
346      -#endif
347      +#undef SC_NO_STDGETS_
348      +
349      +#endif // SYSTEMC_H
```

## Steps

Step1: Build the two PoC libraries

1. Copy the 'msvc10' directory to the SystemC architecture alias 'msvc14' for Microsoft Visual Studio 2015
2. Initialize environment variables  
The project setup uses environment variables to reference SystemC, Boost and CCI directories. These can be set in System Settings or using a Visual Studio launch batch file as shown below:

A screenshot of a Visual Studio launch batch file. The window has a yellow title bar and a menu bar with 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains the following commands:

```
CALL "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat" amd64
set SYSTEMC_HOME=/cygdrive/c/Users/spoddar/tools/systemc-2.3.1
set BOOST_HOME=/cygdrive/c/Users/spoddar/tools/boost/1.57.0/win_64/vc_12/include/boost-1_57
set CCI_HOME=/cygdrive/c/Users/spoddar/GitHub/cci
set CXX=c1START C:\Users\spoddar\tools\cygwin64\Cygwin.bat -l
```

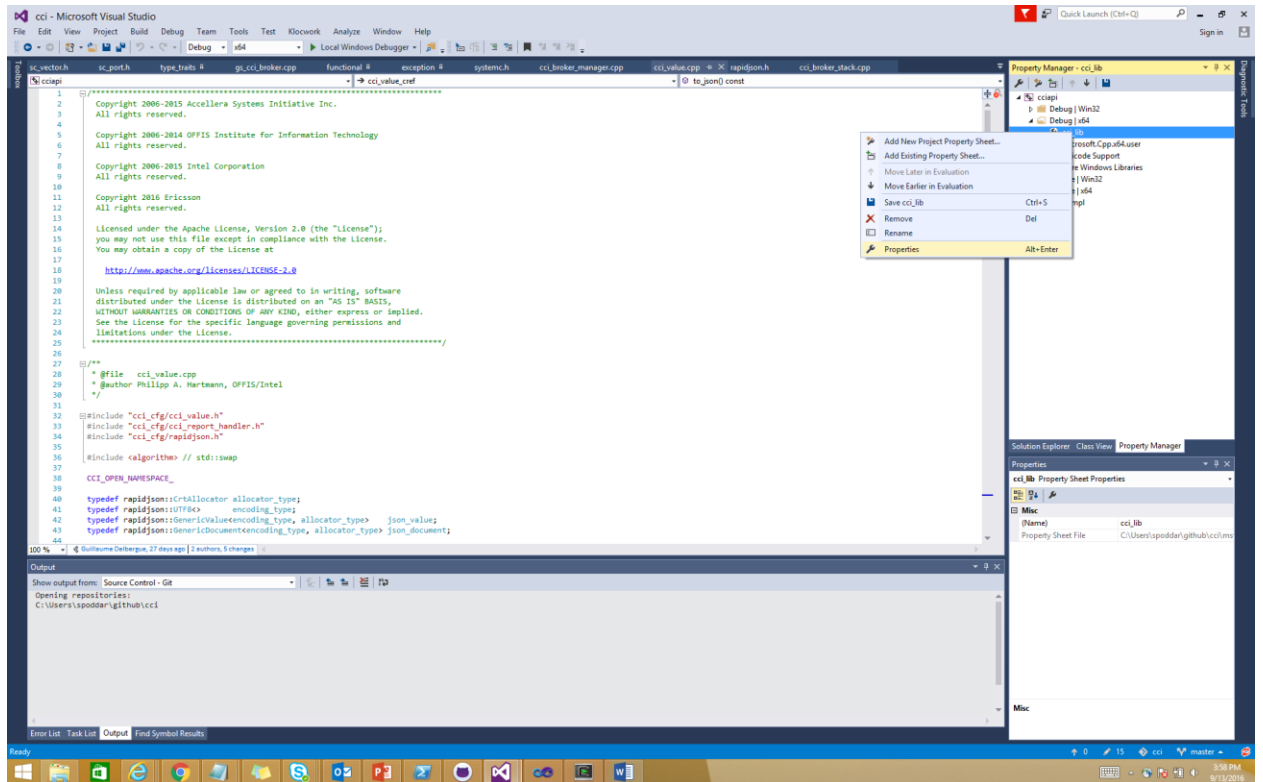
3. Update the CCI libraries property sheet (msvc14/cci/cci\_lib.props)
  - Change the MSVC macro from "msvc10" to "msvc14".

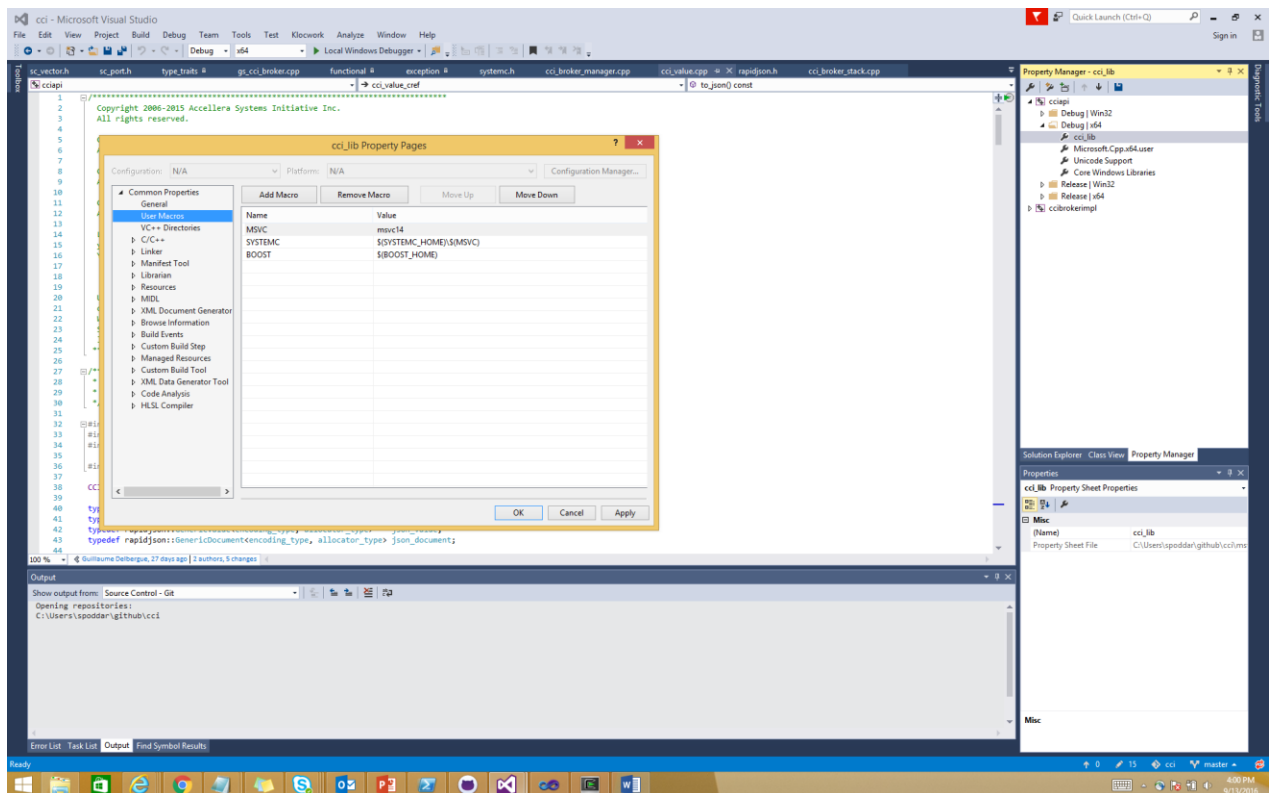
These two projects share this property sheet which specifies compiler and linker paths for SystemC and Boost using environment the variables \$SYSTEMC\_HOME and \$BOOST\_HOME.

Open the existing MSVC Solution

  - File-> Open->Project/Solution
  - From your CCI home -> msvc14 -> cci->cci.sln
  - Under the 'cci.sln' solution, you can see three MSVC Projects:
    - cciapi
    - ccibrokerimpl

You can view the common settings from the Property Manager by right clicking on the cci\_lib property sheet for any of the CCI libraries and selecting "Properties".





#### 4. Build the Solution

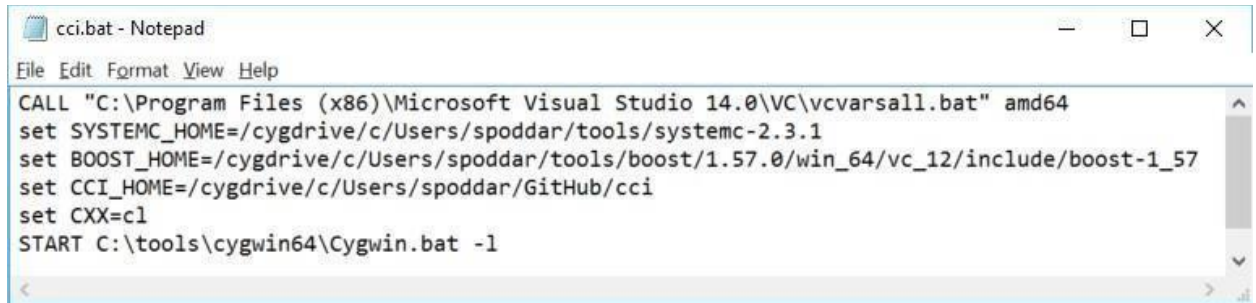
- Choose the configuration
  - o Debug / Release
  - o Win32 / x64
- Build the whole cci solution
- The two CCI PoC libraries will be created for the given configuration.

## Step2: Run regression tests

The UNIX regression script requires a UNIX environment, such as Cygwin.

#### 1. Initialize environment

The key environment variables need to be initialized in a manner consistent with our Visual Studio setup. This can be done in a Cygwin shell launch script as shown below, or simply included in your shell startup settings (i.e. .bashrc).



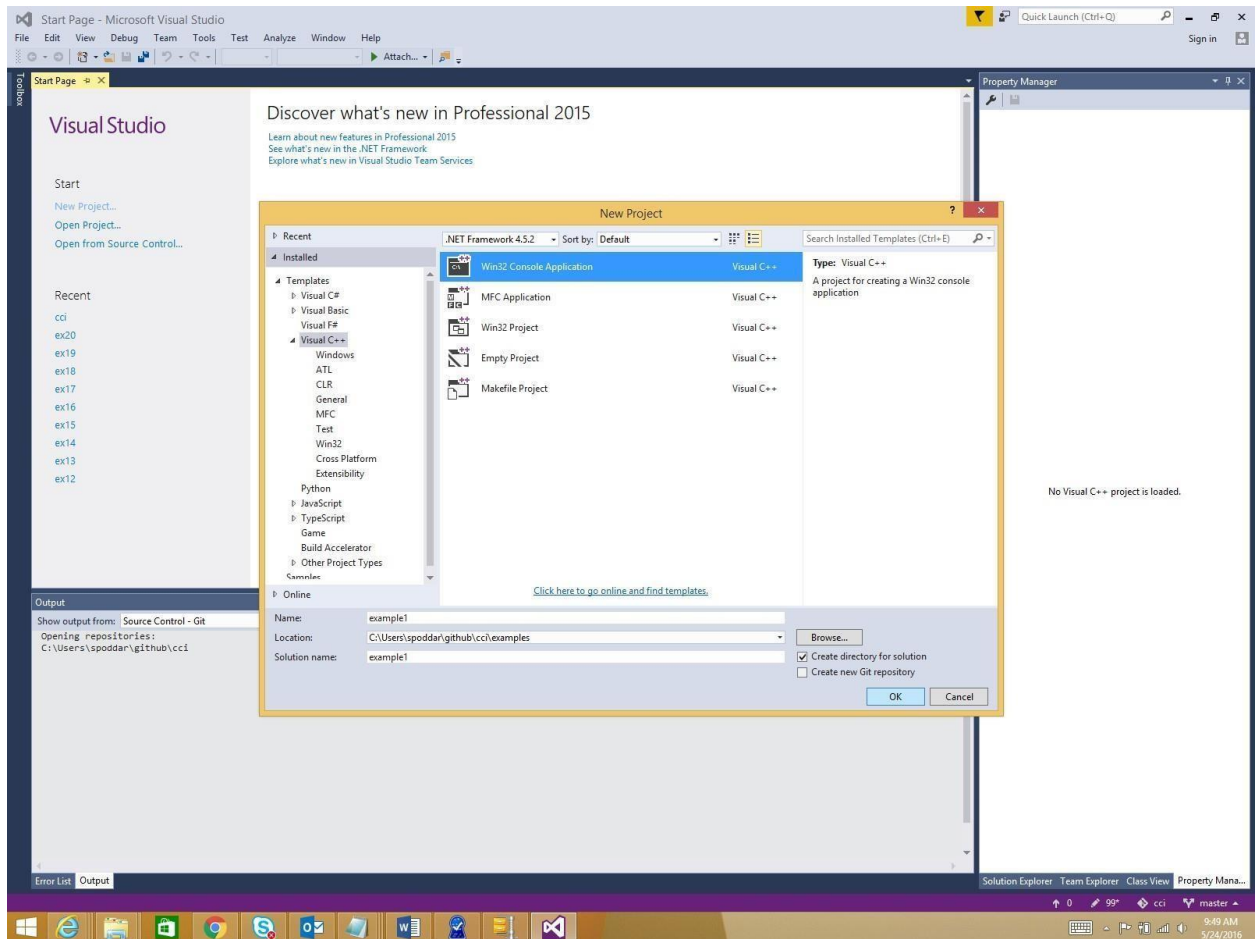
```
cci.bat - Notepad
File Edit Format View Help
CALL "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat" amd64
set SYSTEMC_HOME=/cygdrive/c/Users/spoddar/tools/systemc-2.3.1
set BOOST_HOME=/cygdrive/c/Users/spoddar/tools/boost/1.57.0/win_64/vc_12/include/boost-1_57
set CCI_HOME=/cygdrive/c/Users/spoddar/GitHub/cci
set CXX=c1
START C:\tools\cygwin64\Cygwin.bat -1
```

2. Start a properly initialized Cygwin Shell  
Note: if link.exe is found from the Cygwin installation (which results in linker failures) instead of the Visual Studio installation, a path adjustment will be necessary:  
    \$ PATH=/cygdrive/c/Program\ Files\ \ (x86\)/Mirosoft\ Visual\ Studio\  
14.0/VC/bin/amd64:\$PATH
3. Create a directory (under cci/) to hold the regression results and cd into it  
Ex. mkdir run  
    cd run
4. Run the verify.pl script as described in cci/examples/README.txt  
Ex. ../scripts/verify.pl -g dev\_examples examples (runs all CCI examples using debug library)  
Note: add the verbose flag (-v) to see compiler/linker commands.
5. Compare against the expected results in cci/scripts/results/summary.txt

---

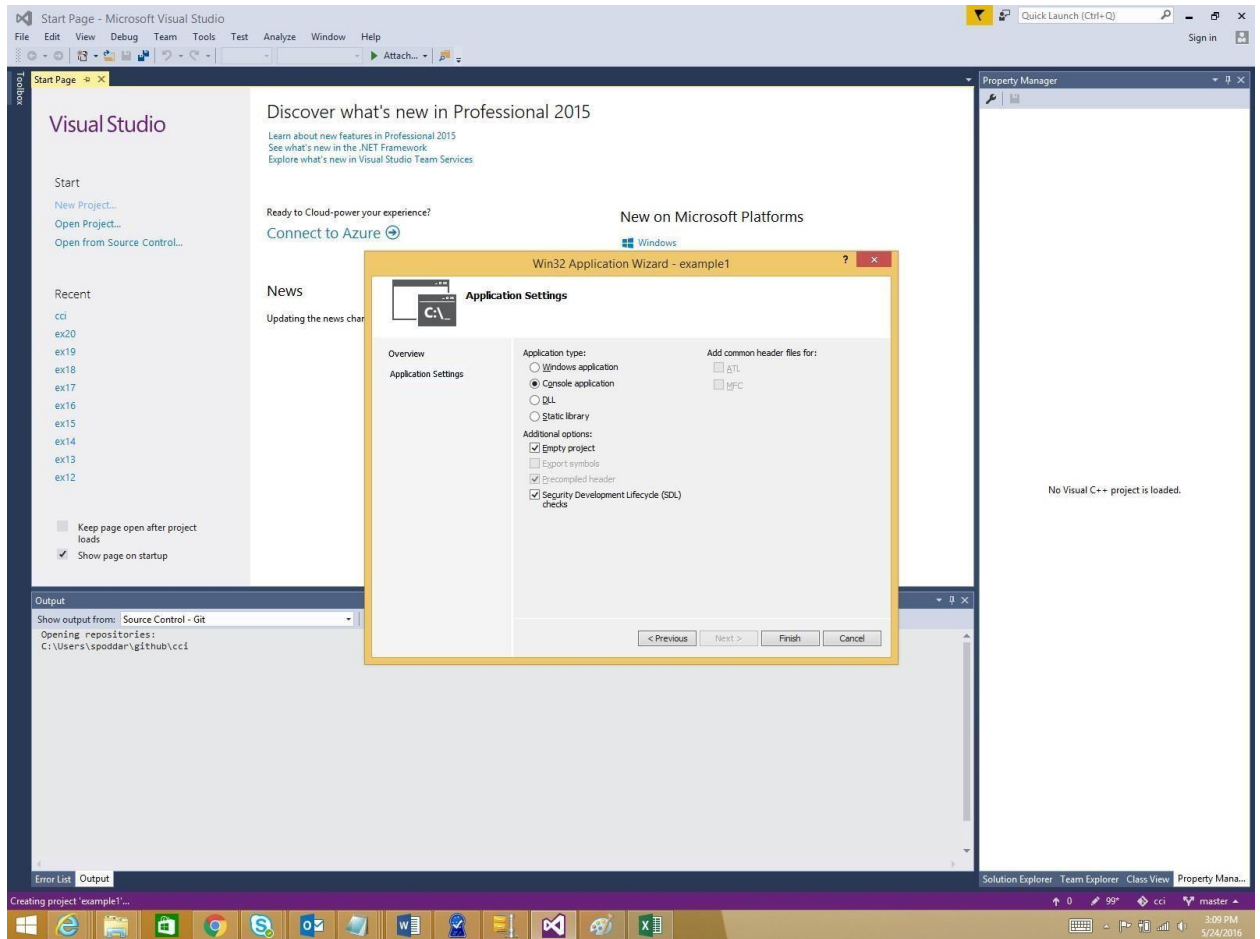
### Step3: Building MSVC Projects for CCI applications

1. Open your Microsoft Visual Studio 2015 cci solution
2. File-> New-> Project  
    Select Visual C++ in the left pane  
    Select Win32 Console Application in the middle pane  
    Name: ex01\_proj  
    Location: C:\Users\spoddar\github\cci\msvc14\cci\ (same as for CCI libs)  
    Solution: Add to solution (to have all CCI projects together)  
    Hit the OK button.

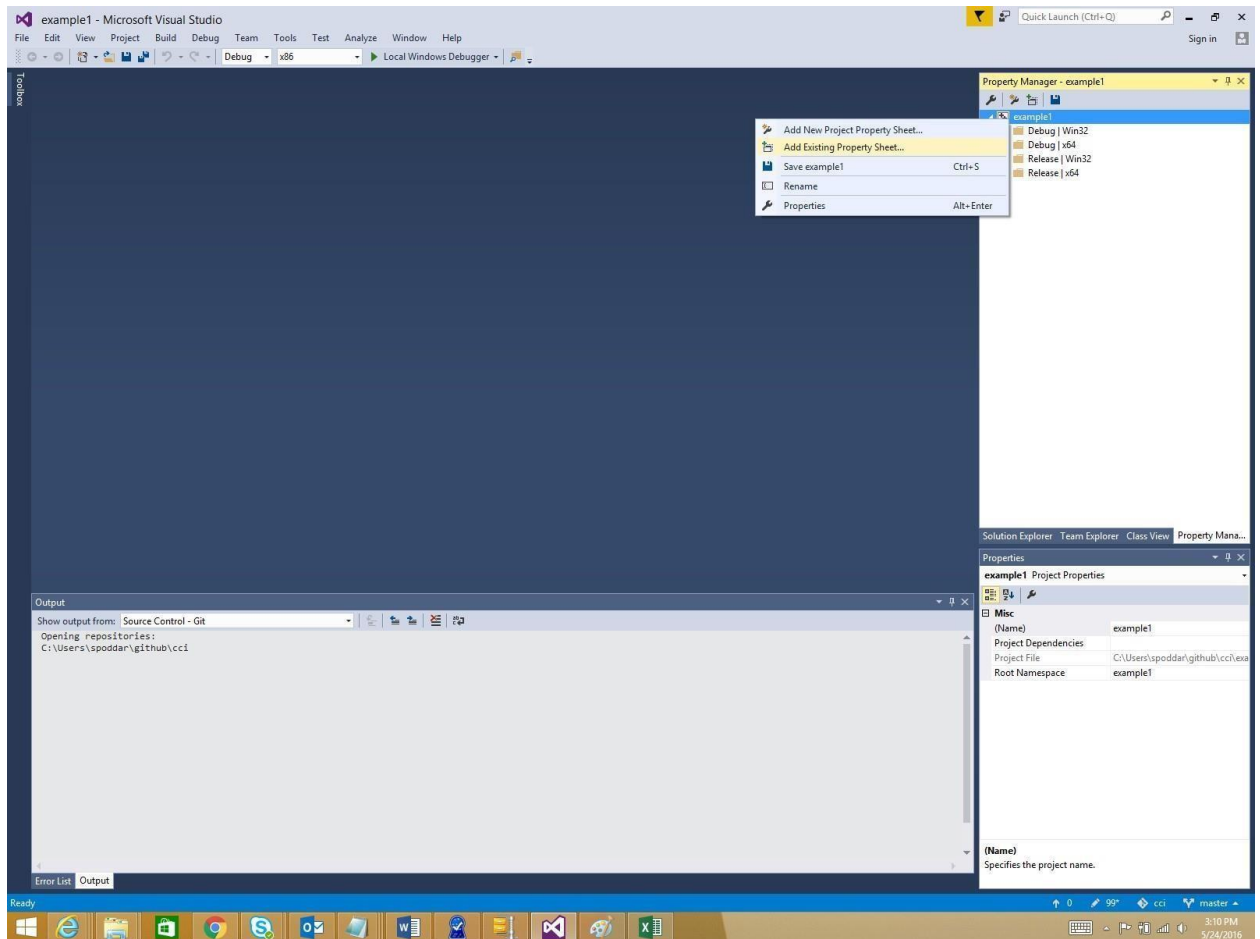


3. Hit Next on the Win32 Application Wizard  
Select Application Type: Console Application  
Additional options: Emptyproject  
*Hit Finish*



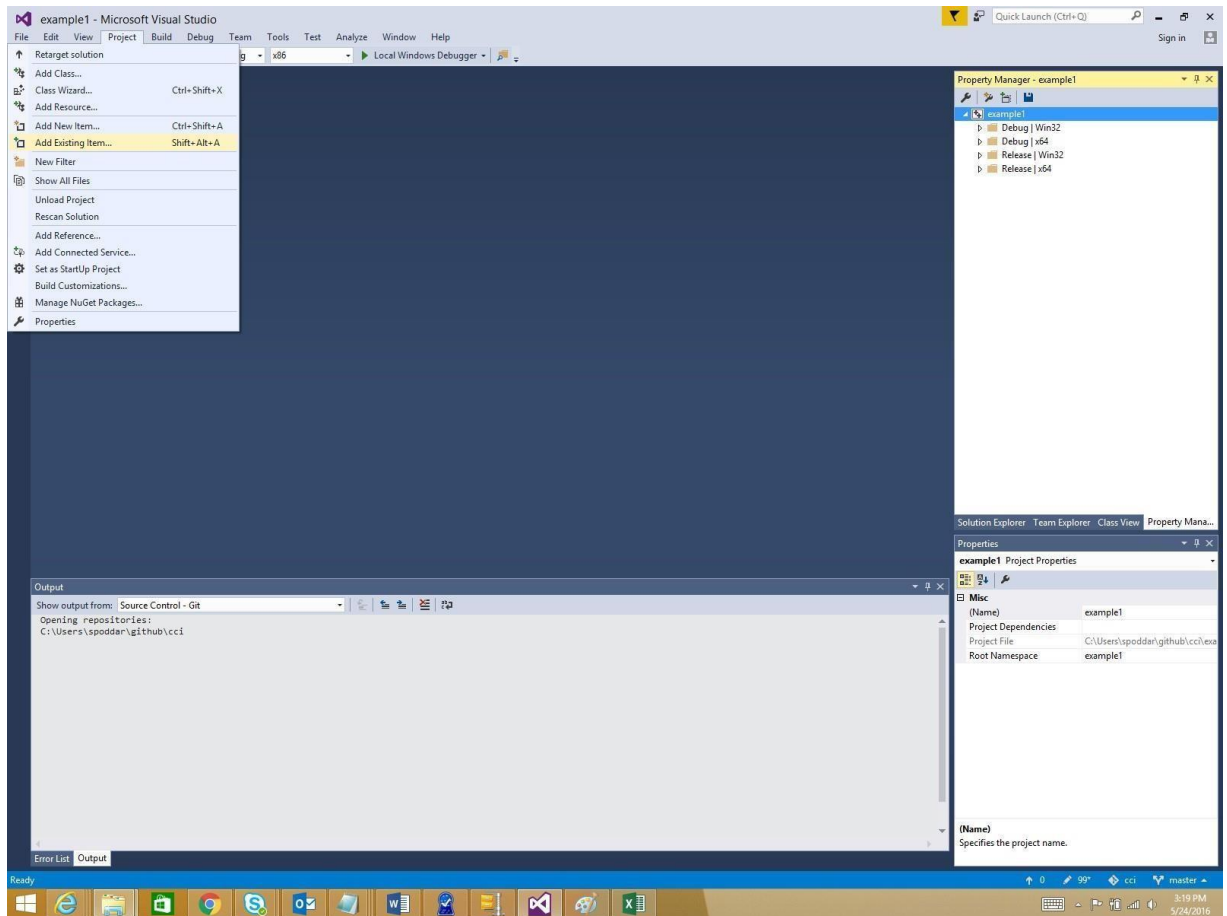


4. Go to Property Manager-> right click example1-> Add existing property sheet  
Select cci\_examples.props from msvc14\cci  
Example C:\Users\spoddar\github\cci\msvc14\cci\cci\_examples.props.

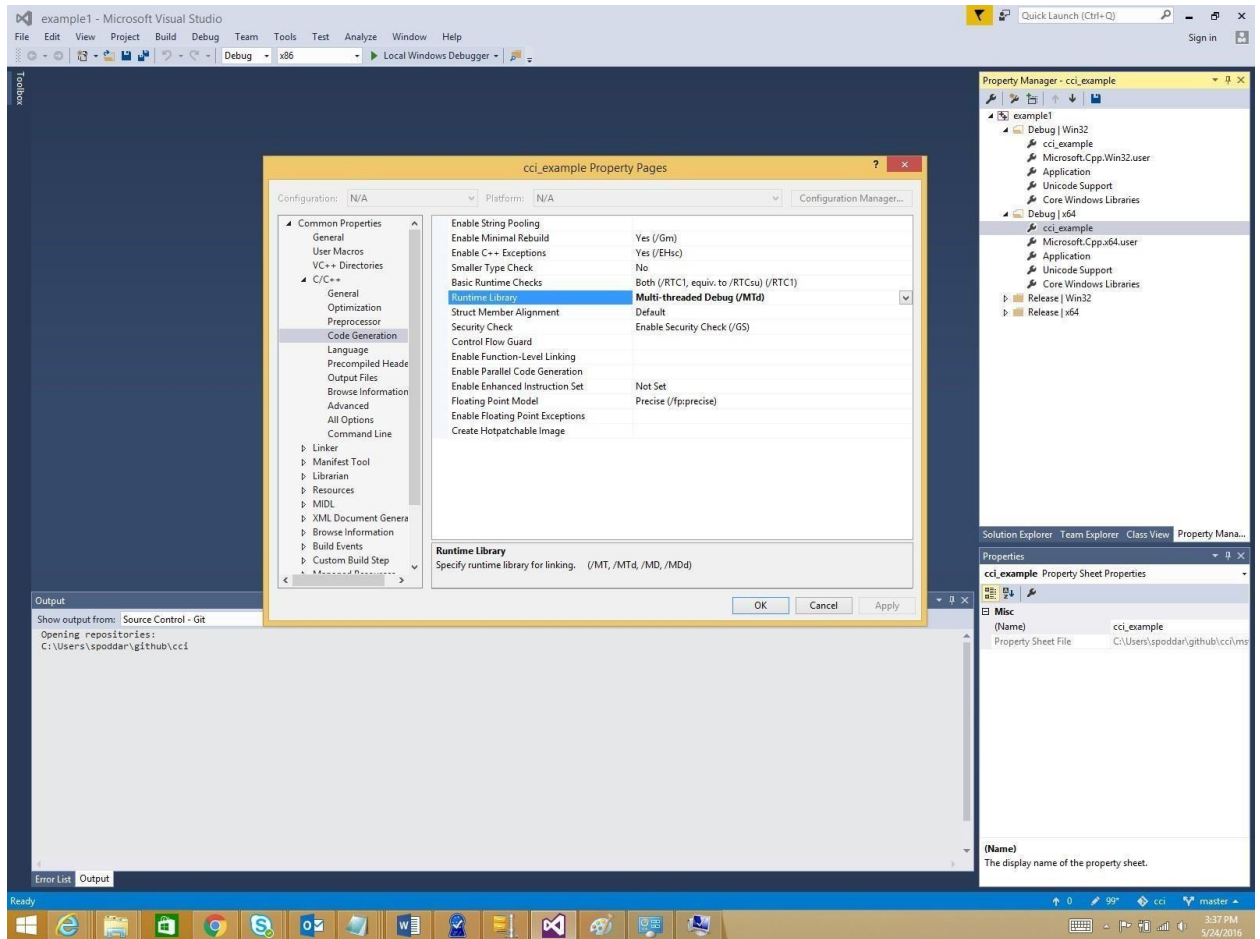


5. Return to Solution Explorer and select Project->Add existing Item
  - a. Navigate to the corresponding example directory (e.g. C:\Users\spoddar\github\cci\examples\ex01\_Simple\_Int\_param) and select all \*.h and \*.cpp files then hit the Add button.

Note: For gs\_example\_diff\_impl among dev\_examples, remember to add all \*.h and \*.cpp files from the subdirectory param\_impl.



- Verify your configuration (e.g. x64-Debug). From the Solution Explorer right click on the project and Select Properties. Navigate to Configuration Properties->C/C++->Code Generation->Runtime Library and choose the same runtime library used to build your SystemC (e.g. /MTd).



You are now ready to build and run/debug.